[0113] Alternatively, the components of the internal web services management system **1600** may be components added to a gateway module **300**, **500**, **900**.

[0114] With another aspect of the gateway module **900**, a mechanism is described for returning a web service contract to a third-party client application developer, which is tailored to that specific developer, who, in turn, can tailor that contract to different end users of that client application **15**. This increased flexibility is made possible by adding a layer above the web services **25**, such that the developer of those web services, hereby referred to as the web service developer, does not need to be concerned with the process of limiting access to third-party developers, who are programmatic consumers of the web services **25**, or to end users. This extra layer, referred to as the gateway module **300**, **500**, **900**, can provide this flexibility at two different points in time, one during development and the other at run-time.

[0115] In the web service commerce model, each web service provider **20** hosts a set of web services **25** either to be consumed over the web by an application publisher, which in turn caters to the end user (i.e., consumer), or another web services provider **20**. The other web service provider **20** may bind to those web services **25** across the Internet, aggregate the web services **25** with their own set of web services **25**, or build upon the web services **25** to provide more sophisticated web services **25** of their own. Therefore, a chain of producer-consumer relationships exists between the client application **15** and suppliers of the lowest level web services **25**. The scenario can be contrasted with the traditional web application model, which is client-server oriented, operating between the web browser on the end user side and an application hosted on a particular web site.

[0116] **FIG. 17** shows an example of a supply chain **1700** of producer-consumer relationships between the client application **15** and web services providers **20**. The supply chain **1700** comprises a client application **15**, web services providers A-E **20**, and web services A1-A9, B1-B8, C1-C5, D1-D7, and E1-E825. The client application **15** uses web services A1, A2, A4, A5, A6 and A825. In order to supply web services A2 and A525, web services provider A uses web services B1 and B325, respectively, from web services provider B **20**. In order to supply web service B125, web services provider B **20** uses web service C425 from web services provider C **20**. In order to supply web service B325, web services provider B **20** uses web service D325 from web services provider D **20**. In order to supply web service C425, web services provider B **20** uses web service E325 from web services provider E **20**.

[0117] Each party in the supply chain **1700** would perceivably have similar concerns. For example, each might want to keep a database of its users, whether private or corporate, and authenticate, authorize and bill them accordingly. Since the technology for hosting of web services **25** is relatively new, currently each entity in the supply chain implements their own business logic to handle the aforementioned concerns.

[0118] The following is a summary of how a web service infrastructure **501**, **1601** provides a chain of producer-consumer relationships, according to an embodiment of the invention. The gateway module **300**, **500**, **900** caters to the common hosting, monitoring and administrative needs of entities in the web service supply chain. This embodiment concerns the manner in which the gateway module **300**, **500**, **900** is architected and deployed. The embodiment includes four features.

[0119] One feature is that the web services infrastructure **201**, **501**, **1601** includes one single logically coherent entity (the gateway module **300**, **500**, **900**) through which communication between client applications **15** and hosted web services **25** are routed. Tightly associated with it is any logic that requires an understanding of how to handle events that occurred within the web services **25**. The centralization of this logic is desirable to provide a comprehensive solution for the web service provider **20**.

[0120] Consider a typical scenario where the provider **20** needs to authenticate and authorize the client **15**, log any events that occur during any access, delegate the request to the appropriate web service **25** as necessary and log any events that occur during the process. At the least, an event infrastructure should be provided to all modules of which the infrastructure is aware, so that events can be sent and that the infrastructure is aware of any module that need to be notified of events. For example, the billing module **970** is notified in the event that a web service **25** is being accessed in order to do its job; and the authorization module **525** might want to notify that a login has failed. The modules should either directly or indirectly be able to communicate with each other. In an implementation where no such centralized infrastructure exists, it is difficult to add modules that need to be notified of events. Thus a comprehensive solution is not be practical without a centralized infrastructure, such as the web services infrastructure **201**, **501**.

[0121] Another feature is that the gateway module **300**, **500**, **900** is able to support off-the-shelf web services **25** as-is without need for adaptation. This is achieved by monitoring low-level requests that comes through, executing necessary logic and finally delegating to the appropriate web service. This is beneficial to both web service providers **20** and web service authors, as they do not have to adapt their logic in order for the services to be hosted, administered and monitored.

[0122] Yet another feature is that the gateway module **300**, **500**, **900** is capable of masking the interfaces, addresses and service description of each web service **25** to appear different to client applications **15**, which are able to access the web services **25** as advertised in a transparent manner.

[0123] The masquerading of interface and service description is desirable to allow the web service provider **20** is to be able to rename web services functionalities and add parameters to their list of formal arguments. This capability is particularly desirable for authentication and authorization purposes. For example, the provider may want to assign the client an authentication ID that once authenticated, requires the client to access web service functions (or methods) with the authentication ID. Here the gateway module **300**, **500**, **900** disguises these web service functions as having an extra authentication ID parameter.

[0124] The masquerading of web service addresses serves the above purposes. Another effect is that it allows the service provider **20** to transparently aggregate services offered by another web service provider **20** as though it is one of its own.

[0125] A fourth feature is that the gateway module **300**, **500**, **900**, along with all the accompanying functionality, is